ARCADA

Pulsmätning och stegräkning från Android Wear

Utveckling av en applikation för att spara mätdata

Fredrik Lund

Examensarbete

Informationsteknik

2016

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informations- och medieteknik
Identifikationsnummer:	4903
Författare:	Fredrik Lund
Arbetets namn:	Pulsmätning och stegräkning från Android Wear - Utveckling av en applikation för att spara mätdata
Handledare (Arcada):	Magnus Westerlund
Uppdragsgivare:	

Sammandrag:
Detta examensarbete beskriver utvecklingen av en applikation för att överföra data från en Android Wear smartklocka till en Android smarttelefon som sedan skickar datan vidare till en server. Servern sätter in datan i en MySQL-databas. Android Wear smartklockornas sensorer undersöks även. Smartklockorna har sensorer för stegräkning och pulsmätning. I examensarbetet beskrivs det hur sensorerna fungerar. Därtill genomgås Androids historia i korta drag samt basfunktionerna i Android operativsystemet och Android Wear operativsystemet. Dessutom jämförs några Android Wear-smarklockor och Android utvecklingsmiljöerna Eclipse och Android Studio beskrivs. Examensarbetet består av planering och programmering av en applikation till Android och planering, programmering och installation av server/databas-delen.

Nyckelord:	Android, Android Wear, Android Applikation, REST, JSON, Databas
Sidantal:	39
Språk:	Svenska
Datum för godkännande:	31.5.2016

DEGREE THESIS	
Arcada	
Degree Programme:	Information and Media Technology
Identification number:	4903
Author:	Fredrik Lund
Title:	Heart rate monitoring and step counting with Android Wear Development of an application to store measured data
Supervisor (Arcada):	Magnus Westerlund
Commissioned by:	

Abstract:
This thesis work describes the development of an application to transfer data from an Android Wear smartwatch to an Android smartphone, which then sends the data to a server. The server puts the data into a MySQL database. The sensors of Android Wear smartwatches are also investigated. Smartwatches have sensors for step counting and heart rate measurement. The thesis work describes how the sensors work. Additionally a short run through Android's history and the basic functions of the Android operating system, and Android Wear operating system. Moreover the thesis work compares some Android Wear smartwatches and describes two Android development environments: Eclipse and Android Studio. The work consists of planning and programming of an application for Android and planning, programming and installation of server / database part/section.

Keywords:	Android, Android Wear, Android Application, REST, JSON, Databas
Number of pages:	39
Language:	Swedish
Date of acceptance:	31.5.2016

INNEHÅLL

Figurer

Tabeller

Förkortningar

AJAJ	Asynchronous JavaScript And JSON
AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
APK	Android Application Package
ARM	Acorn RISC Machine
ART	Android Run Time
CORBA	Common Object Request Broker Architecture
ECG	Electrocardiography
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines Corporation
IDE	Integrated Development Environment
JIT	Just In Time
JSON	JavaScript Object Notation
LED	Light Emitting Diod
mAh	milliAmpere-hour
MB	Megabyte
PPG	Photoplethysmogram
REST	Representational State Transfer
RPC	Remote Procedure Call
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WAR	Web application archive
XML	Extensible Markup Language

1 INLEDNING

Android Wear är en smartklocka, vilket betyder att den är en klocka som kommunicerar med en smarttelefon (Android eller iPhone). I detta arbete använder jag en Android smarttelefon. Enligt Cody Willard kommer marknaden för så kallade wearables (på svenska, kroppsnära teknik), som är en term för alla slags smarta produkter som bärs på kroppen, att växa. [1] Jag tror också att Android Wear och även smartkläder kommer att bli vanligare i framtiden, så jag tänkte att det skulle vara intressant att se vad man kan göra med denna första generation av Android Wear smartklockor. Jag undersökte olika slags smartklockor och valde Android Wear för att den hade inbyggd steg- och pulsmätare medan de andra smartklockorna i min jämförelse hade endast stegmätare, vilket gör att man måste anskaffa extra utrustning om man vill göra pulsmätningar. Motorola, LG och en del av Samsung smartklockorna har samma operativsystem, Android Wear, de andra Samsung smartklockorna har operativsystemet Tizen. Övriga tillverkare av smartklockor som Suunto, Polar och Fitbit har sina egna operativsystem. Detta gör att Android Wear smartklockorna har möjlighet att bli den största plattformen och således skapa det största ekosystemet. Android Wear uppdateras med jämna mellanrum och för tillfället är version 1.4 lanserad. Uppdateringen kan ha både positiva och negativa sidor: det kan komma ny funktionalitet till Android Wear, men det kan också hända att det med nya uppdateringar tas bort någon funktionalitet som används i en applikation.

1.1 Syfte och mål

Målet med detta examensarbete är att skapa en applikation som samlar upp data ur Android Wear smartklockornas sensorer och lagrar datan i en databas, samt att undersöka vad man kan få ut för data från sensorerna.

Frågor som skall besvaras:

- Vilka sensorer finns det?
- Hur får man data ur Android Wear smartklockans sensorer?
- Hur skapar man och upprätthåller kommunikationen mellan Android smarttelefon och Android Wear smartklocka?

1.2 Metod

Jag började projektet med att jämföra smartklockor som fanns på marknaden i början av år 2015, jag använde två olika utvecklingsmiljöer den ena, Android Studio för applikationsdelen och den andra, Eclipse för serverdelen. Jag skapade en applikationsprototyp för kommunikation mellan smartklocka, smarttelefon och server. När jag stötte på problem sökte jag information på Internet hur andra hade löst liknande problem och tillämpade dem i min prototyp. Då jag hade en fungerande prototyp testade jag att den fungerade i praktiken.

1.3 Avgränsning

Jag avgränsar mitt arbete till att använda Android produkter för applikationsdelen och se vad jag kan få ut för data ur Android Wear smartklockans sensorer. Jag fokuserar mig på smartklockans sensorer och deras funktionalitet och undersöker inte Android Wears andra funktionaliteter. Jag utvecklar en prototypsapplikation och jag tar inte ställning till olika säkerhetsaspekter för kommunikationen mellan Android smarttelefonen och servern. Den data som jag fått sätter jag in i en MySQL databas som jag har byggt upp. Databasen behöver inte fungera i andra sammanhang utan är uppbyggd endast för detta projekt. Jag valde att arbeta med en Linuxbaserad server och en MySQL databas eftersom de är bekanta för mig.

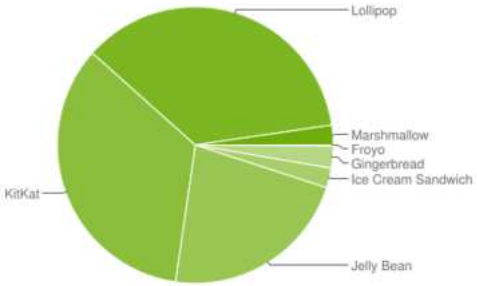
2 MJUKVARA OCH HÅRDVARA

2.1 Android OS

Android, Inc. grundades av Andy Rubin, Rich Miner, Nick Sears och Chris White i Palo Alto i Kalifornien i oktober 2003. Google köpte Android Inc den 17 augusti 2005. [2] Android 1.0 kom ut den 23 september 2008, den 22 oktober 2008 kom den första smarttelefonen med Android (HTC Dream) ut på marknaden. Android 1.5 som kom ut den 27 april 2009 och hade kodnamnet Cupcake var den första Android versionen med kodnamn. Alla versioner från och med 1.5 har haft sötsaksnamn som kodnamn. Den nyaste versionen av Android är 6.0.1 Marshmallow och den kom ut den 9 december 2015. [3]

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.3%
4.1.x	Jelly Bean	16	8.1%
4.2.x		17	11.0%
4.3		18	3.2%
4.4	KitKat	19	34.3%
5.0	Lollipop	21	16.9%
5.1		22	19.2%
6.0	Marshmallow	23	2.3%

Data collected during a 7-day period ending on March 7, 2016.
Any versions with less than 0.1% distribution are not shown.



Figur 1. Den procentuella andelen de olika versionerna av Android som var i användning i mars 2016. [4]

Android OS är ett mobiloperativsystem baserat på Linux-kärnan, men Android är inte en version av Linux på samma sätt som andra distributioner som t.ex. Redhat, Debian eller Ubuntu. Android saknar vissa delar som finns i Linux, men den har också delar som inte finns i Linux. Android har t.ex. förbättrad strömhantering, vilket är viktigt för mobila enheter, en väldigt snabb interprocesskommunikation och en mekanism för sandboxing, så att de olika applikationerna är isolerade från varandra. [5, s. 33]

Sandboxing betyder att en applikation startas skilt från andra applikationer och får en viss mängd minne att arbeta med, applikationen får inte heller fulla rättigheter att ändra filer på smarttelefonen applikationen körs på eller att accessera filer som andra applikationer skapat.

Google har gjort en basversion av Android, men p.g.a. att Android är modifierbar så ser Android litet olika ut i olika smarttelefoner när tillverkarna kan sätta in egen programkod och egna program. Motorolas Android och LGs Android kan se lite olika ut och ha olika funktionaliteter men båda har ändå samma bas.

Datorer kopplade till ett nätaggregat behöver inte vara sparsamma med energiförbrukningen på samma sätt och de använder Java Virtuell Maskin, som är en "one size fits all" lösning. Eftersom Android är mobilt måste man beakta batterikonsumtionen och därför har en annan Virtuell Maskin (VM) vid namn Dalvik skapats för Android. [5, s. 36]

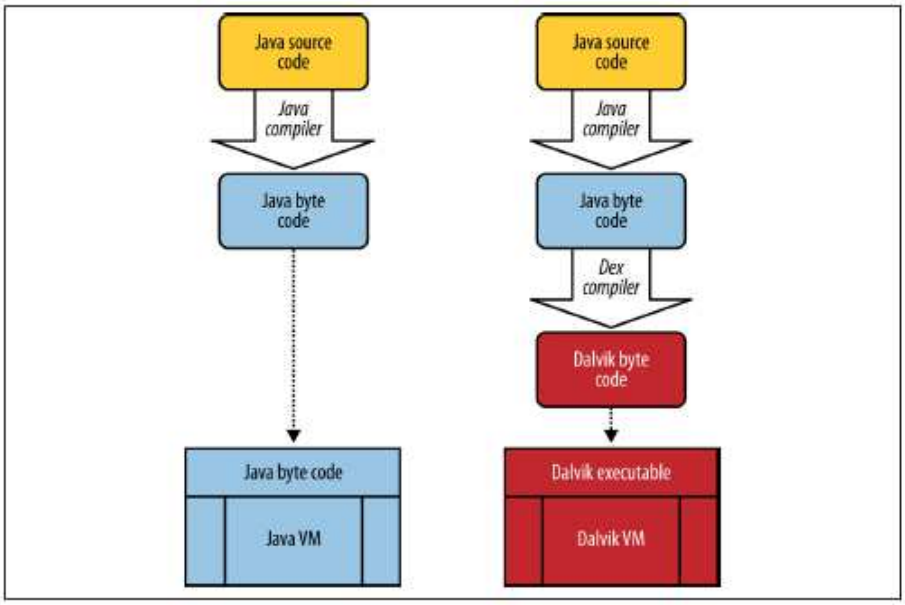
En virtuell maskin (VM) är en dator som inte finns fysiskt, endast virtuellt. Alltså är VM ett program som emulerar som en fysisk dator, men i verkligheten bara är ett program. Virtuella maskiner emulerar en vald dator arkitektur och funktioner av riktiga eller hypotetiska datorer. För att implementera dessa kan det behövas specialiserad hårdvara eller mjukvara eller båda. [6]

Dalvik är en virtuell maskin för att köra applikationer även kallad applikationsvirtualisering och körs som en vanlig applikation utanpå operativsystemet, Android OS i detta fall. Varje Dalvik VM stöder körandet av endast en applikation åt gången och Dalvik skapas när applikationen startar och förstörs när applikationen stängs, detta innebär att flera Dalvik Virtual Maskiner kan vara aktiva samtidigt. En viktig del i Androids säkerhetsmodell är sandboxing, som sker när det finns flera Dalvik Virtuell Maskiner igång samtidigt. Applikationsvirtualisering innebär också att programmeringsomgivningen blir icke-plattformbunden dvs. att man inte behöver fundera på hårdvaran eller operativsystemet, utan programmet körs på samma sätt på alla slags plattformer. [6]

Dalvik VM är uppbyggd så att den skall ta så lite utrymme som möjligt för att spara minnesanvändningen bl.a. att den delar på systembibliotek istället för att skapa en ny kopia varje gång den behöver ett visst systembibliotek. I Dalvik laddas data inte lika mycket till RAM-minnet som på datorer för att RAM-minnet är mindre på smarttelefoner än på datorer och för att datorer har hårddiskar som ibland är långsamma på att läsa och skriva medan smarttelefoner alltid har solid state storage vilket gör dem mycket snabbare att läsa och skriva. Bara den data som behöver ändras sätts in i RAM-minnet på smarttelefonen. [5, s. 37]

Att skriva kod för en Android applikation och en Java applikation är i praktiken lika, båda skrivs i språket Java, men kompilerandet och Virtual Maskinerna är olika. I en

Java applikation måste Java-källkodsfiler kompileras till Java byte-kod och sedan köras i en Java VM, medan för en Android applikation måste Java källkodsfiler också kompileras till Java byte-kod men sedan ännu kompileras vidare av en Dex kompilator till Dalvik byte-kod för att till sist köras i en Dalvik VM. [5, s. 37]



Figur 2. Illustration av standard Java kompilation till vänster och Dalvik kompilation till höger.[5]

Dalvik använder så kallad trace-based just-in-time (JIT) (på svenska, "precis i rätt tid" kompilering), som kom med i Android 2.2 (Froyo). JIT optimerar körningen av applikationer genom att profilera applikationer varje gång de körs och dynamiskt kompilera segment som körs ofta från byte-kod till nativ maskin programkod. Dalvik tolkar resten av applikationens byte-kod, och gör så kallade traces, vilka gör signifikanta förbättringar i prestandan genom att nativ-köra korta byte-kod segment. [7]

Från och med Android 5.0 (Lollipop) har Dalvik slopats, nu används Android Runtime (ART) som är en "application runtime environment". ART använder "ahead-of-time (AOT)" "i förtid" kompilation. AOT kompilerar hela programmet till nativ maskin kod när programmet installeras. Genom att eliminera Dalviks tolkning och "trace-based JIT" kompilation förbättrar ART körningseffektiviteten och minskar strömförbrukningen, vilket förbättrar batterilivslängden. På samma gång försnabbar ART körningen av ap-

plikationer, förbättrar minnesallokeringen och skräpinsamlings mekanismerna, har nya debug verktyg och mera precisa hög nivå profileringar av applikationer. [7]

För att upprätthålla bakåtkompatibilitet använder ART samma byte-kod som Dalvik, genom samma standard .dex filer som del av APK (Android application package) filer, medan .odex filer byts ut mot Executable and Linkable Format (ELF) "körbart och länkbart format". När en applikation är kompilerad med ART:s "on-device" dex2oat "utility", körs den enbart från den kompilerade ELF exekverbara filen. Som ett resultat av detta eliminerar ART en del överlopps kod associerade med Dalviks tolkning och "trace-based JIT" kompilation. Det negativa med ART är att det kräver extra tid för kompilationen när en applikation installeras och att applikationerna tar upp lite mera utrymme för att spara den kompilerade programkoden. [7]

2.2 Android Wear

Android Wear är ett operativsystem för smartklockor som grundar sig på Android OS. Det presenterades vid Google I/O konferens tillfället den 18 mars 2014, då gavs också ett test program för programutvecklare. Vid ett annat tillfälle den 25 juni 2014 lanserades Samsung Gear Live och LG G Watch.

Smartklockan måste vara ansluten till en Android smarttelefon via Bluetooth för att all funktionalitet skall fungera och för att den skall få all information den behöver för applikationer eftersom själva smartklockan inte har en egen internetanslutning. För att kunna para ihop en Android smarttelefon med en Android Wear smartklocka måste smarttelefonen ha Android version 4.3 eller högre.

Android Wear behöver ett eget användargränssnitt eftersom den är en smartklocka och har annan funktionalitet och annat användningssätt än en smarttelefon. Därför skiljer sig Android Wear operativsystemet ganska mycket från Android operativsystemet. Google har skapat ett nytt operativsystem och användargränssnitt för Android Wear. Användargränssnittets två nya huvuddelar är Context Stream och Cue Card. [8, s. 341]

2.2.1 Context Stream

Android Wear OS är ett kortbaserat system, vilket innebär att användargränssnittet är en vertikal ström av kort som visas automatiskt baserat på tid, plats och många andra användaraktiviteter och intressen. [8, s. 342] Varje kort i context streamen kan innehålla ytterligare sidor av relevant information. Användare kan svepa vertikalt för att navigera igenom korten i context streamen och svepa horisontellt för att se mera information. När användaren vill göra sig av med ett kort i context streamen kan det tas bort med att svepa från höger till vänster. Applikationer kan sätta upp sina egna kort till context streamen, vilket tillåter användare att ta emot information utan att behöva uttryckligen öppna programmet som skapat kortet till context streamen. Applikationer kan också öppnas via context streamen vid behov.

Man kan utveckla helskärmsläges applikationer för Android Wear, men Google rekommenderar att man utvecklar applikationer som man enkelt kan integrera med context stream. Detta innebär att en av utgångspunkterna när man utvecklar för Android Wear är att lära sig att integrera de kort som genereras av ens applikation i den existerande Android Wear context streamen. [8, s. 342]

2.2.2 Cue Card

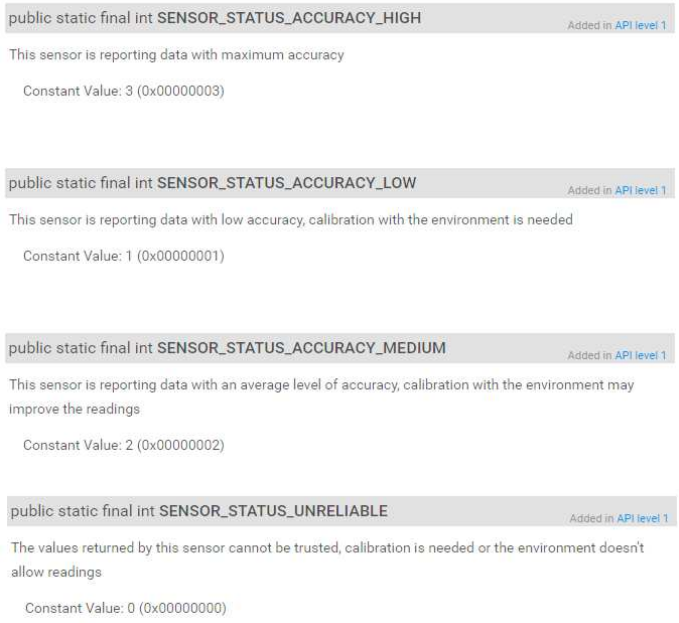
Cue Card dvs. kommando kort uppfattar röstkommandon och söker efter en applikation som kan matchas till begärt kommando och startar sedan den valda applikationen.
Kommandot skall börja med "Okay, Google" sedan säger man den handling som man vill att smartklockan eller den anslutna smarttelefonen skall utföra. T.ex. "Okay, Google", "Take a note", "buy apples and oranges" eller "Okay, Google", "Send Bob an email", och sedan diktera man vad det skall stå i e-posten. [8, s. 341-342]

2.3 Vad de olika smartklockorna kan mäta

I detta kapitel kommer jag att gå igenom vilken data de olika sensorerna i Android Wear smartklockorna kan ge ut och hur jag kom åt denna data.

2.3.1 Vilken data Android Wear ger ut

För att få reda på i vilken datatyp man får ut värdena ur Android Wear sensorerna läste jag på Androids API (Application Programming Interface, applikationsprogrammeringsgränssnitt) dokumentation [9]. Enligt Sensor Event webbsidan [10] ges noggrannheten (Accuracy) i datatypen public int och enligt Sensor Manager webbsidan [11] ges noggrannheten i fyra olika värden mellan 0 och 3. Värdet 0 står för opålitlig data, 1 för låg noggrannhet, värdet 2 för medelbra noggrannhet och värdet 3 för hög noggrannhet.



Figur 3. Sensor noggrannhetsvärden [11]

Enligt Sensor Event webbsidan [10] ges värdet (value) som själva sensorn, i detta fall hjärtpulsmätaren och stegräknaren i datatypen public final float[].

Fields		
public int	accuracy	The accuracy of this event.
public Sensor	sensor	The sensor that generated this event.
public long	timestamp	The time in nanosecond at which the event happened
public final float[]	values	The length and contents of the `values` array depends on which `sensor` type is being monitored (see also `SensorEvent` for a definition of the coordinate system used).

Figur 4. Vilken data man kan få ut ur SensorEvent [8]

På Sensor Event webbsidan [10] kan man läsa en hel del om hur olika sensorers värden bestäms, men inte hur heart rate värdet bestäms. Jag hittade inte heller på developer-webbsidorna [9] över huvudtaget hur Android Wear sensorerna bestämmer noggrannheten på datan.

2.4 Jämförelse av olika Android Wear smartklockor

I detta delkapitel kommer jag att jämföra de olika smartklockorna som fanns på marknaden då jag valde en smartklocka för detta projekt. Jag jämför vilka sensorer de har, batteristorlek och skärmstorlek för att kunna bestämma batterilivslängd, ju större skärm desto kortare livslängd som tumregel.

2.4.1 Motorola Moto 360

Motorola Moto 360 har en 9-axis sensor pedometer för att räkna tagna steg och en PPG sensor för att mäta pulsen. Den har ett batteri på 320 mAh och en skärmstorlek på 320x290. [12]

Tabell 1. Tabell över Motorola Moto 360 tekniska specifikation

Batteri	320mAh
Skärmstorlek	320x290 205ppi.
Processor	TI OMAP 3 processor 1 GHz
RAM	512MB
Sensorer:	
Puls	PPG
Steg	9-axis sensor pedometer

2.4.2 LG G Watch R

LG G Watch R har en 9-axis sensor pedometer för att räkna tagna steg och en PPG sensor för att mäta pulsen. Den har ett batteri på 410 mAh och en skärmstorlek på 320x320. [13]

Tabell 2. Tabell över LG G Watch R tekniska specifikation

Batteri	410mAh
Skärmstorlek	1.3-inch Full Circle P-OLED (320 x 320)
Processor	Qualcomm® Snapdragon™ 400, 1.2GHz
RAM	512MB
Sensorer:	
Puls	PPG
Steg	9-Axis (Gyro, Accelerometer, Compass)

2.4.3 Samsung Gear Live

Samsung Gear Live har en Accelerometer och Gyroscope för att räkna tagna steg och en PPG sensor för att mäta pulsen. Den har ett batteri på 300 mAh och en skärmstorlek på 1.63”. [14]

Tabell 3. Tabell över Samsung Gear Live tekniska specifikation

Batteri	300mAh
Skärmstorlek	1.63” Super AMOLED
Processor	???
RAM	512MB
Sensorer:	

Puls	PPG
Steg	Compass, Accelerometer, Gyroscope

2.4.4 Vilken smartklocka valdes och varför

Alla smartklockorna har liknande sensorer och deras skärmar är så pass lika stora att jag ansåg att den lilla skillnaden i skärm storlek var obetydlig. Dessutom har alla smartklockorna lika mycket RAM. Jag valde LG G Watch R för att den har det största batteriet.

3 PULS- OCH STEGMÄTNING

3.1 Pulsmätning

Det finns primärt två olika sätt att mäta en människas puls, ECG (Electrocardiography) och PPG (Photoplethysmogram). ECG mäter en persons puls med hjälp av elektroder som fästs på kroppen. Dessa elektroder mäter små elektriska förändringar i huden som uppstår när hjärtmuskeln pumpar. [15] PPG igen mäter en persons puls genom att belysa huden och mäta förändringen i mängden ljus som reflekteras tillbaka. [16]

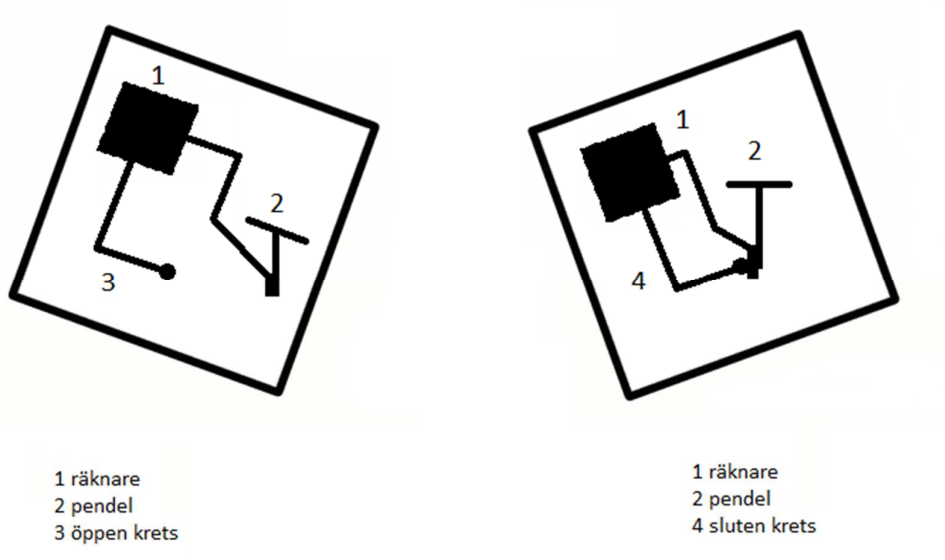
3.1.1 Pulsmätning med Android Wear

Android Wear använder sig av Photoplethysmogram (PPG) för att mäta användarens puls.

Med varje hjärtslag pumpas blod ut i kroppen, pulstrycket dämpas något när det når huden, men är ändå tillräckligt för att tänja ut artärerna och arterioler i underhudsvävnaden och registreras av PPG mätaren. Pulsmätningen sker genom att belysa huden med en LED och mäta mängden ljus som reflekteras till en fotodiod. Mängden ljus som reflekteras ändras beroende på volymförändringen som orsakas av tryckpulsen som kommer från ett hjärtslag. Varje hjärtslag visas som en topp. Man skall inte sätta smartklockan för spänt på armen för då komprimeras huden och det kan uppstå felmätningar.

3.2 Stegmätning

När man går vaggar kroppen från sida till sida och detta använder en pedometer för att räkna steg. En elektronisk pedometer fungerar så att när man är stilla är pedometerns krets öppen och ingen ström flödar igenom den, men när man tar ett steg så rör en pendel på sig och slår fast så att det blir en sluten krets och ström flödar igenom kretsen och räknar det som ett steg och sedan far pendeln tillbaka till sin neutrala position och det blir en öppen krets igen tills man tar ett steg till och processen börjar om igen. [17] En vanlig design av pedometer med en elektrisk krets som alternerande blir öppen och sluten när man går. När pedometern lutar åt vänster så sluts kretsen och ett steg räknas av processorn som i Figur 5 markeras med 1. Sedan när den lutar till höger igen så blir det en öppen krets och är färdig att räkna ett nytt steg.



Figur 5. En vanlig design av pedometer med en elektrisk krets.[17]

4 KOMMUNIKATION MELLAN KOMPONENTERNA

4.1 Använda utvecklingsmiljöer

För att skapa en Android applikation behöver man inte ha en Andriod smarttelefon, det räcker att ha Android SDK (Software Development Kit) och en utvecklingsmiljö, t.ex. Eclipse eller Android Studio.

I detta delkapitel kommer jag att beskriva utvecklingsmiljöerna Eclipse och Android Studio som finns för Android.

4.1.1 Eclipse

Eclipse SDK är en gratis programvara med öppen källkod. Den är en integrerad utvecklingsmiljö (integrated development environment, förkortat IDE) för programmering, som har plugins så att man kan programmera i många olika språk bl.a. C, C++, JavaScript, PHP och Python.

Eclipse skapades år 2001 av ett litet team av experter på IBM för att företaget ville minska på det stora antalet inkompatibla utvecklingsmiljöer som det sålde och höja återanvändningen av vanliga komponenter i dessa utvecklingsmiljöer. Eclipse utvecklades från en lång rad av utvecklingsmiljöer, börjande med IBM VisualAge för Smalltalk och IBM VisualAge för Java utvecklades till IBM VisualAge Micro Edition. Denna produkt var det första allvarliga experimentet att skriva hela IDE i Java. Många koncept som fanns i IBM VisualAge Micro Edition används i Eclipse. IBM VisualAge Micro Edition svaghet var att det var svårt för tredje parter att skapa nya komponenter till IBM VisualAge Micro Edition för att den var monolitisk och hade stängd källkod, samt att den inte var designad för att vara komponent öppen. Eclipse är designat med tanke på att vara plug-in vänligt och Öppen källkod så att det är lättare att vidareutveckla den för både Eclipse Organisationen och andra utvecklare. I januari 2004 skapades Eclipse Foundation som en oberoende icke vinstdrivande stiftelse som förtroendeorgan för Eclipse community. [18]

4.1.2 Android Studio

Android Studio presenterades på Google I/O konferensen i Moscone Center i San Francisco den 16 maj 2013 av Googles produkt manager Ellie Powers. Den första stabila versionen (1.0) av Android Studio, kom ut i december 2014.

Jag använde Android Studio för att skapa applikationsdelen av projektet, för att jag ville testa på någonting nytt och för att jag inte har varit riktigt nöjd med Eclipse. Jag var

nöjd med mitt val för att Android Studio är lite mera logisk och för att emulatorn var snabbare än i Eclipse.

Googles mål med Android Studio är att det skall bli det program Android utvecklare använder för att skapa nya och fortsätta utveckla gamla applikationer. Google lovade att Android Studio skall ersätta Eclipse genom att Android Studio gör utvecklare mer produktiva, de har också gjort en migrationsguide för att visa hur man flyttar ett projekt från Eclipse till Android Studio. Android Studio innehåller mallar och Googles kodexempel, vilket är bra om man inte vill börja med tomma projekt. Android Studio använder ett Gradle-baserat byggsystem som enligt Google ger mycket flexibilitet och stor utvidgningsmöjlighet och möjligheten att bygga i eller utanför IDE. Huvudegenskaper i Android Studio är stöd för så kallade build variationer så att man bättre kan hantera olika typers build (debug eller utgiven) eller olika versioner av samma applikation, gratisversion eller en avgiftsbelagd version. Med Android Studio kan man editera och förhandsgranska sina Android layouts över flera olika skärmstorlekar, språk, och till och med API versioner. Detta är till stor hjälp om man skall utveckla en applikation som kommer att användas i flera Android versioner. Med Android Studio kan man också behandla strängöversättningar för alla sina applikationer. Man kan lätt få fram och analysera alla hård kodade strängar i vilket projekt som helst. [19]

4.2 Serverprogrammering

I detta delkapitel kommer jag att gå genom hur skapandet och implementerandet av serverdelen gick till. Jag kommer också att behandla de delar av min kod som jag själv tycker att var intressanta. Min serverapplikation var implementerad på en Linux server i Arcadas interna nät.

4.2.1 Programmeringen

Jag använde Eclipse för att skapa programkod för servern och Tomcat 7.0 för att emulera en server. Den emulerade servern var till för att testa om serverkoden jag har skrivit fungerar som planerat. Man måste ha en emulerad smarttelefon för att kunna kommuni-

cera med den emulerade servern, smarttelefonen emulerade jag med Android Studios smarttelefon emulator.

Apache Tomcat är en öppen källkod implementation av olika Java teknologier som t.ex. Java Servlet och JavaServer Pages. Apache Tomcat utvecklas i en öppen och medverkande miljö och är utgiven under Apache License version 2. [20] Den används i en mängd storskaliga och viktiga webb applikationer av ett flertal olika industrier, företag och organisationer som t.ex. General Motors och WalMart i USA och Hotels and Accomodations i Australien. [21]

Serverapplikationen har endast två klasser DataEntry (DE) och DataResource (DR) Figur 6. DR tar emot data och bestämmer vad DE skall göra med den. Datan som DE tar emot formas om till en MySql-sats så att man antingen får data eller sätter in data i databasen. Serverapplikationen baserar sig på **Re**presentational **S**tate **T**ransfer, (REST) som stöder sig på ett cachebart klient-server kommunikationsprotokoll. REST är en arkitekturstil för nätverksapplikation. För att sköta kommunikationen används en simpel HTTP kommunikation i stället för att använda komplexa mekanismer som t.ex. SOAP. REST använder fyra simpla HTTP förfrågningar för att erhålla eller editera data; @GET få data, @POST skapa data, @PUT skapa eller uppdatera data och @DELETE ta bort data. På detta sätt använder REST HTTPs alla fyra CRUD (Create/Read/Update/Delete) data operationer. Fastän REST är ett simpelt protokoll så kan det utföra samma funktioner som andra Web Services utför. Dock är REST inte en standard och det kommer inte att finnas en W3C (World Wide Web Consortium) rekommendation för REST. [22]

```
@Path("/data")
public class DataResource {
    @GET
    @Path("{namn}")
    @Produces(MediaType.APPLICATION_JSON)
    public DataEntry getDataEntry(@PathParam("namn") String name)
    {
        return new DataEntry(name);
    }
    @POST
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.TEXT_HTML)
    public String addDataEntry(DataEntry newEntry)
    {
        newEntry.saveEntry();
        return "Success!";
    }
    @PUT
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.TEXT_HTML)
    public String editDataEntry(DataEntry newEntry)
    {
        newEntry.editEntry();
        return "Success!";
    }
    @DELETE
    @Path("{name}")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.TEXT_HTML)
    public String deleteEntry(@PathParam("name") String name)
    {
        DataEntry newEntry = new DataEntry();
        newEntry.deleteEntry(name);
        return "Success!";
    }
}
```

Figur 6. DataResource

JSON som behandlas nedan används i applikationen för att överföra data mellan smart-telefonen och servern se Figur 6 och Figur 17. JSON, (JavaScript Object Notation) är ett öppen standard format som använder text som är läsbar för människor för att överföra dataobjekt bestående av attribut-värde par. JSON är det huvudsakliga dataformatet som används för asynkronisk browser/server kommunikation (AJAJ) vilket motsvarar XML (använt av AJAX). [23]

Fastän JSON ursprungligen kom från JavaScript är det ett språkoberoende dataformat. Kod som skall analysera eller generera JSON är ett textformat som är fullständigt språkoberoende, men använder liknande syntax som C-familjens språk såsom C, C++,

C#, Java, JavaScript, Perl och Python. JSON baserar sig på en version av JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. [21]

Ett objekt är ett osorterat set av attribut-värde par, ett objekt börjar med { (vänster klammer) och slutar med } (höger klammer). Varje attributnamn följs av ett ':' (kolon-tecken) och värdet, varje namn/värde par är åtskiljt av ett ',' (kommatecken) ex:{Namn:Kalle, Namn:Pelle}. [23]

Databasen där mätdatan lagrades har två tabeller, en för användare (User) och en för data (Data). Users innehåller namn, lösenord, ålder, längd, vikt, genomsnittlig hjärtpuls. Data innehåller namn, datum, tid, hjärtpuls noggrannhet, hjärtpuls och tagna steg. Tabellerna är länkade med varandra med namn kolumnen.

Figur 7. Databasens tabeller, hur de är länkade

Som nämnts i kapitel 4.1.1 har Eclipse-plugins. Jag använder en av dessa, WTP (Web Tool Platform), med vilken man kan skapa och exportera server moduler. Server moduler exporteras i form av WAR filer. Jag använde denna plugin för att skapa server delen av mitt arbete (REST tjänsten). WAR står för Web application ARchive. En WAR fil används för att distribuera en samling av JavaServer Pages, Java Servlets, statiska webbsidor och andra resurser som tillsammans bildar en webbapplikation.

4.2.2 Sätta upp servern

I detta delkapitel går jag igenom hur jag satte upp WAR filen på servern i Arcadas interna nät. Jag kommer inte att beskriva hur man kommer in på Arcadas interna nät eftersom jag anser att det inte är viktig information för detta arbete.

Servern jag använde är en Linux Ubuntu. Jag började med att installera Java JRE på servern för Apache Tomcat 7 kräver minimum Java 6 JDK. Jag installerade Java 7 JDK med kommandot:

```
sudo apt-get install openjdk-7-jre-headless
```

Figur 8. Kommando för att installera jdk7

För att jag skulle kunna sätta in min WAR fil installerade jag Apache Tomcat 7.0.52 genom att först ladda ner den med det första kommandot i Figur 9 till mappen Downloads, efter det packade upp jag paketet med det andra kommandot och med det tredje kommandot flyttade jag "apache-tomcat-7.0.52" mappen från "Downloads" mappen till "etc" mappen på servern.

I Figur 9 kan man läsa kommandona jag använde för att installera Tomcat på servern.

```
wget http://mirror.cc.columbia.edu/pub/software/apache/
tomcat/tomcat-7/v7.0.52/bin/apache-tomcat-7.0.52.tar.gz

tar xvzf apache-tomcat-7.0.52.tar.gz

mv apache-tomcat-7.0.52 /etc/apache-tomcat-7.0.52
```

Figur 9. Tomcat installeringskommandon

Som följande installerade jag Tomcat modulen "mod_jk" för att ansluta Tomcat servlet containern med Apache genom kommandot:

```
apt-get install libapache2-mod-jk
```

Figur 10. Kommando för att installera libapache2-mod jk

Efter det kollade jag att Tomcat har en " listening connector " för inkommande ajp13 förfrågningar från "mod_jk connectorn" genom att granska att filen "server.xml" i "etc/apache-tomcat-7.0.52/conf/" inte har bort kommenterat detta:

```
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

Figur 11. Connector

Till nästa skapade jag en fil "jk.conf" i "/etc/apache2/mods-enabled/"

med följande innehåll:

```
<IfModule jk_module>
JkWorkersFile /etc/libapache2-mod-jk/workers.properties
JkLogFile /var/log/apache2/mod_jk.log JkLogLevel info
JkShmFile /var/log/apache2/jk-runtime-status
JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "
JkMount /DataService/* ajp13_worker
JkMount /DataService ajp13_worker
</IfModule>
```

Figur 12. jk.conf innehåll

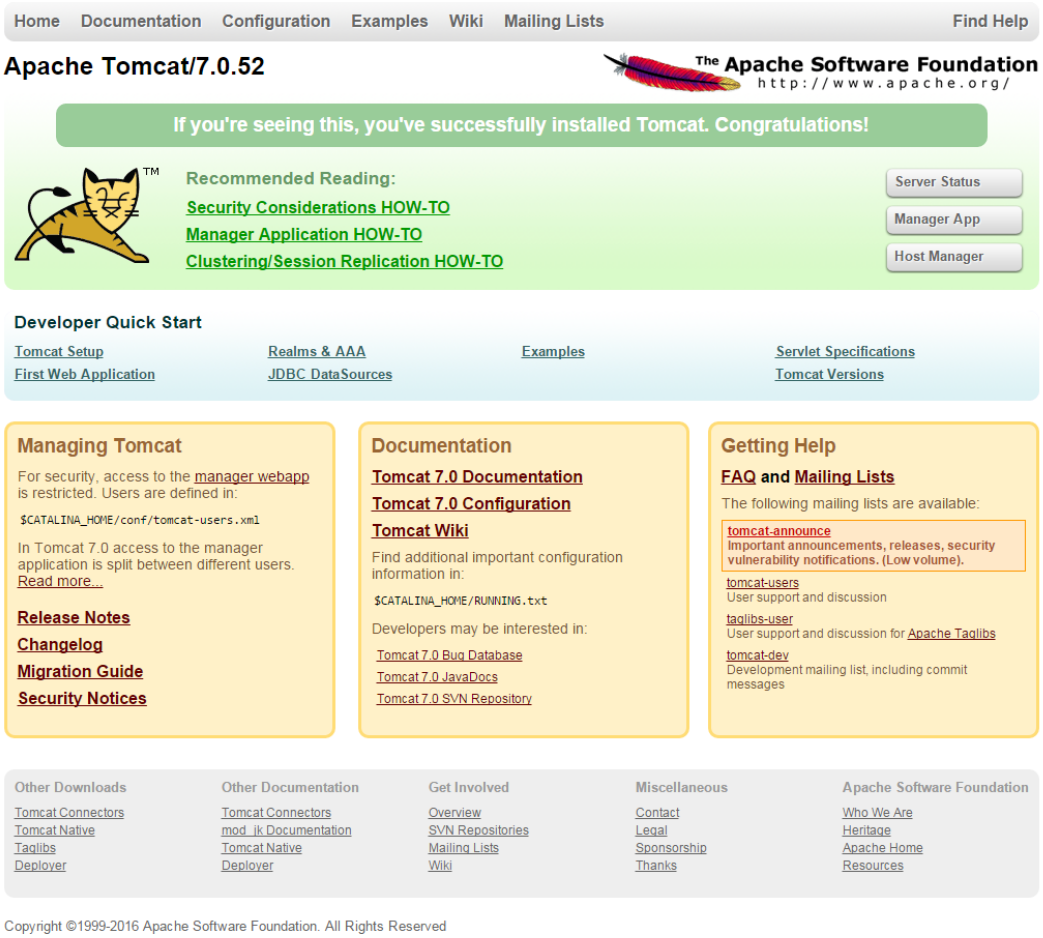
Sedan måste man konfigurera "tomcat-users.xml" filen i ”/etc/apache-tomcat-7.0.52/conf/” genom att sätta till raderna som finns i Figur 13 mellan <tomcat-users> och </tomcat-users> så att man kommer åt Tomcat Web Application Manager för att ladda upp sin WAR fil.

```
<role rolename="manager-gui"/>
<user username="User" password="Password" roles="manager-gui"/>
```

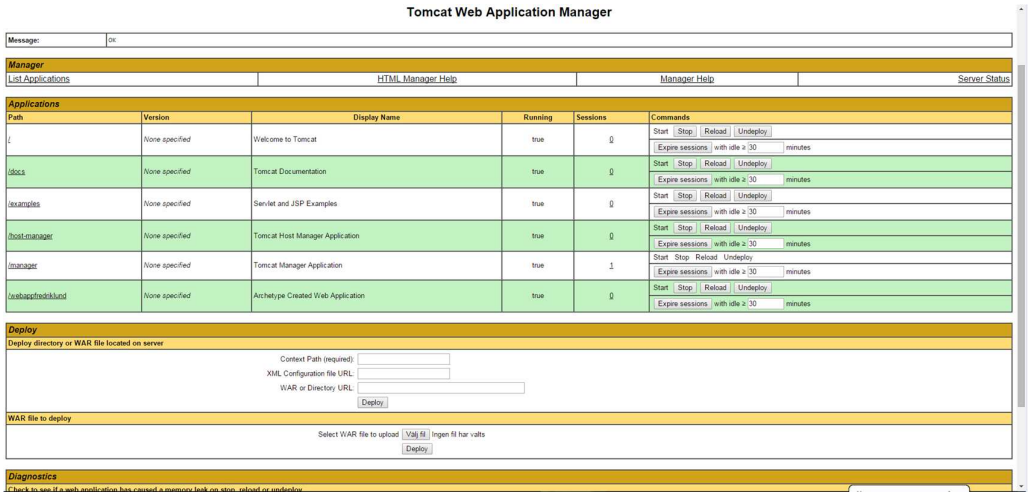
Figur 13. "Raderna som behövs" för att skapa en Tomcat manager-gui

Sedan startade jag om Apache-servern med "apachectl restart" och startade om Tomcat servern med kommandona: etc/ apache-tomcat-7.0.52/bin/shutdown.sh och etc/ apache-tomcat-7.0.52/bin/startup.sh

För att komma åt min Tomcat server skrev jag serverns ip-adress på Arcadas interna nät och satte till porten :8080 i webbrowserns adressfält, t.ex. 192.168.0.1:8080. Figur 14 nedan visar sidan. När man klickar på Manager App knappen kommer man till en sida där man kan ladda upp en WAR fil.



Figur 14. Tomcat servers Front page



Figur 15. Tomcats web application Manager sida

Sedan klickar man på knappen "Välj fil" i "WAR file deploy" rutan och väljer sin WAR fil som man har exporterat från Eclipse. Efter detta är ens server modul implementerad och färdig för att användas, i detta fall att ta emot data och sätta in den i en databas.

För mitt projekt behövde jag en databas. Jag valde MySQL för att jag är van att arbeta med den. Första steget var att installera MySQL på servern, detta gjordes med första kommandot i Figur 16. Efter det måste man skapa en databas för att innehålla de två tabellerna som behövs för projektet det gjorde jag med det andra kommandot, tredje och fjärde kommandona är för att skapa "Data" och "Users" tabellerna.

```
sudo apt-get install mysql-server

CREATE DATABASE DataDb;

USE DataDb;

CREATE TABLE Users (Namn varchar(20) PRIMARY KEY,Password varchar(20),Age int
(3), Length int(3),Weight int(3),HeartRateAVG int(3));

CREATE TABLE Data (Namn varchar(20),Date DATE,Time TIME,HeartRateAccuracy int
(1),HeartRate int(3),StepCounter int(6),
FOREIGN KEY (Namn) REFERENCES Users(Namn));
```

Figur 16. MySQL installeringskommandon

4.3 Intressanta delar av applikationen

I följande delkapitel kommer jag att gå igenom olika metoder och klasser som jag använde i mitt arbete och som jag tycker att är intressanta eller viktiga.

4.3.1 Name Checker

Eftersom applikationen skall kunna användas av flera personer och det inte får finnas flera användare med samma användarnamn måste det kontrolleras om ett namn är tillgängligt eller inte. Detta görs med name checker metoden.

Name checker metoden (Figur 17) tar namnet som skall kontrolleras om det är tillgängligt från editName, skickar det till RestGetNew metoden (Figur 18) som skickar en förfrågan till servern och sedan ger tillbaka det som ett svar. Svaret kommer tillbaka som ett JSON-objekt varifrån man tar ut name ur JSON-objektet och kontrollerar om det är available eller något annat. Om det är available som kommer tillbaka kan namnet an-

vändas. Då körs DataEntryNew för att skapa ett konto. Om available inte kommer tillbaka meddelar programmet "Use another name" använd ett annat namn.

```
    EditText searchView = (EditText) findViewById(R.id.editName);
    String response = new RestGetNew
    (this).execute(searchView.getText().toString()).get();
    JSONObject json = new JSONObject(response);
    String tryname = json.getString("name");
    mTextViewError.setText(tryname);
    System.out.println(tryname);
    if(tryname.contains("available")) {
        new RestPostNew(this).execute
        (new DataEntryNew(name,password,age,length,weight,heartrateAVG));
        mTextViewError.setText("User created");
    }
    else
    {
        mTextViewError.setText("Use another name" );
    }
}
else
mTextViewError.setText("please fill in all forms");
```

Figur 17. Name checker metoden i smarttelefon

RestGetNew metoden skickar en förfrågan till servern som sedan skapar ett svar som tar emot svaret från servern och returneras sedan till name checker metoden.

```
public class RestGetNew extends AsyncTask <String, Void, String> {

    NewMember n;

    public RestGetNew(NewMember n)
    {
        this.n=n;
    }

    @Override
    protected String doInBackground(String... sName)
    {
        HttpClient hc = new DefaultHttpClient();
        System.out.println("searching for: "+sName[0]);
        sName[0]=sName[0].replaceAll(" ","%20");
        System.out.println(sName[0]);
        HttpGet request = new HttpGet
        ("http://172.***.***.***:8080/DataService/rest/data/'" + sName[0]+"'");
        ResponseHandler <String> h = new BasicResponseHandler();
        String response = new String();
        System.out.println(request.toString());
        try
        {
            response = hc.execute(request, h);
        }
        catch (Exception e)
        {
            System.out.println(e.getMessage());
        }


        return response;
    }


}
```

Figur 18. RestGetNew metoden

DataEntry metoden kontrollerar om ett namn redan finns i databasen eller om namnet är tillgängligt. Stringen "name" har ”available” som förinställt värde, om namnet man kontrollerar är användbart ändras inte stringen "name", men om namnet redan finns i databasen ändras stringen "name" till ”not”. Detta kan man utläsa i Figur 19. Stringen ”name” skickas sedan som svar till smarttelefonens applikation som sedan kontrollerar om "available" eller "not" kommer tillbaka. Detta kan man utläsa i Figur 17.

Ändringen av string "name" "available","not" uppnås genom att ha resultSet.next() i en if-sats, om databasen inte har namnet man söker blir resultSet.next() "null" inne i if-satsen och if-satsen körs inte. Stringen ”name” ändras inte heller. Men om namnet finns i databasen blir resultSet.next() "true" och if-satsen körs och stringen ”name” ändras från "available" till "not".

```
public DataEntry(String searchName)
{
    try
    {
        Class.forName("com.mysql.jdbc.Driver");
        System.out.println("searching for "+searchName );
        String URI = "jdbc:mysql://172.***.***.***lol/DataDB?user=*****&password=****";

        Connection connection = (Connection) DriverManager.getConnection(URI);
        Statement statement = (Statement) connection.createStatement();

        searchName=searchName.replaceAll("%20"," ");
        String query = "select * from Users where Namn=" + searchName;

        ResultSet resultSet = statement.executeQuery(query);

        name="available";
        if(resultSet.next())
        {
            name = "not";
        }
    }
    catch(Exception e)
    {
        System.out.println("ERROR: " + e.getMessage());
    }

}
```

Figur 19. DataEntry metoden på server sidan

4.3.2 Sensor data och data send

För att man skall kunna ta upp data från Android Wear smartklockans sensorer måste applikationen ha rättigheter till smartklockans kropps sensorer, dessa fås genom att sätta till en "uses-permission" (användnings rättigheter)sats i Android Wear applikationens manifest som i Figur 20.

```
<uses-permission android:name="android.permission.BODY_SENSORS" />
```

Figur 20. Body sensors permission

Metoden onSensorChanged aktiveras när applikationen upptäcker att värdet i någon av Android Wear smartklockans sensorer ändras. Metoden börjar med att kontrollera om det var hjärtpuls mätarens eller stegräknarens värde som ändrades. Sedan skapas ett meddelande som innehåller typ av data: "heart rate" eller "step count", själva datan från sensorn och klockslaget angivet i timmar, minuter och sekunder (t.ex. 12:20:30) det upptäcktes att datan ändrades. Detta meddelande skickas vidare till ett objekt SendToDataLayerThread för att skicka det till smarttelefonen, som sedan skickar meddelandet vidare till servern som sätter in värdena i Data-tabellen. Detta illustreras i Figur 21.

```
public void onSensorChanged(SensorEvent sensorEvent) {
    if (sensorEvent.sensor.getType() == Sensor.TYPE_HEART_RATE) {
        String message = "HeartRate accuracy: " + sensorEvent.accuracy
                + "  " + sensorEvent.values[0]+ " Time: " + currentTimeStr();
        new SendToDataLayerThread("/message_path", message).start();
        Log.d(TAG, message);
    }
    else if (sensorEvent.sensor.getType() == Sensor.TYPE_STEP_COUNTER) {
        String message = "Step Count: " + (int)sensorEvent.values[0]+ " Time: "
                + currentTimeStr()+"       ";
        new SendToDataLayerThread("/message_path", message).start();
        Log.d(TAG, message);
    }
}
```

Figur 21. onSensorChange metoden

För att smartklockan och smarttelefonen skall kunna kommunicera behöver applikationen veta vilken nod smartklockan är ansluten till, detta görs med SendToDataLayerThread objektet. När SendToDataLayerThread vet vilken nod smartklockan är ansluten till skickar smartklockan data i form av ett meddelande till den noden. Anslutningen mellan smarttelefonen och smartklockan har skapats när man har anslutit smartklockan till smarttelefonen.

```
class SendToDataLayerThread extends Thread {
    String path;
    String message;

    // Constructor to send a message to the data layer
    SendToDataLayerThread(String p, String msg) {
        path = p;
        message = msg;
    }

    public void run() {
        NodeApi.GetConnectedNodesResult nodes =
                Wearable.NodeApi.getConnectedNodes(googleClient).await();
        for (Node node : nodes.getNodes()) {
            MessageApi.SendMessageResult result =
                    Wearable.MessageApi.sendMessage(googleClient,
                            node.getId(), path, message.getBytes()).await();
            if (result.getStatus().isSuccess()) {
                Log.v("myTag", "Message: {" + message + "} sent to: " +
                        node.getDisplayName());
            } else {
                // Log an error
                Log.v("myTag", "ERROR: failed to send Message");
            }
        }
    }
}
```

Figur 22. Sendtodatalayer objekt

4.3.3 Datainmatning

Metoden saveEntry är ganska simpel. Den används antingen för att skapa en ny användare till User-tabellen eller för att sätta in data i Data-tabellen. Metoden består av endast en if-else-sats som kontrollerar om meddelandet som servern tar emot innehåller "date". Om meddelandet innehåller "date" så antar programmet att man vill sätta in data i Data-tabellen, om meddelandet inte innehåller "date" så antar programmet att man vill sätta data i User-tabellen. Detta illustreras i Figur 23.

```
public void saveEntry()
{
    try
    {
        Class.forName("com.mysql.jdbc.Driver");
        String URI = "jdbc:mysql://172.***.***.***/DataDB?user=*****&password=****";
        Connection connection = (Connection) DriverManager.getConnection(URI);
        if(date!=null)
        {
            String query = "insert into Data values (\"" + name + "\", \"" + date + "\", \"" +
            time + "\", \"" + hra+ "\", \"" + hr + "\", \"" + sc + "\")" ;
            PreparedStatement ps = (PreparedStatement) connection.prepareStatement(query);
            ps.executeUpdate();
        }

        else
        {
            String query = "insert into Users values (\"" + name + "\", \"" + password + "\", \""
            + age + "\", \"" + length + "\", \"" + weight + "\", \"" + hr + "\")" ;
            PreparedStatement ps = (PreparedStatement) connection.prepareStatement(query);
            ps.executeUpdate();
        }
    }
    catch(Exception e)
    {
        System.out.println(e.getMessage());
    }
}
```

Figur 23. Save Entry metoden

4.3.4 Wakelock

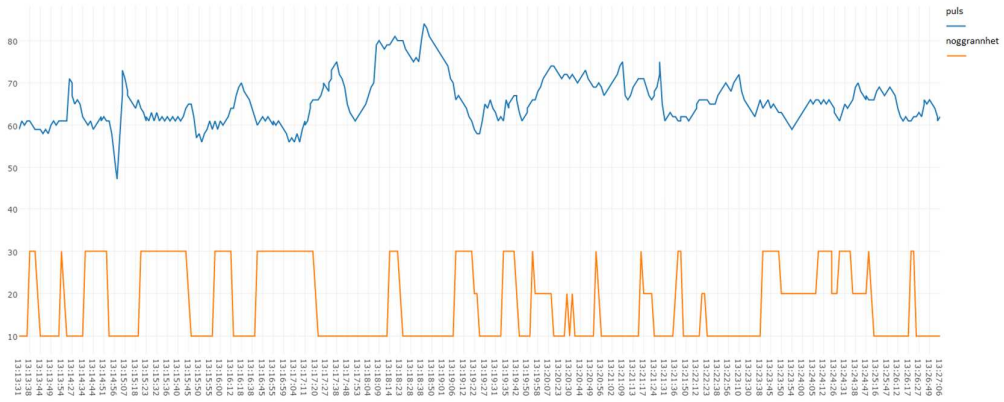
Det största problemet som uppstod var i Android Wear delen av applikationen. Den fungerade inte som jag hade tänkt eftersom när man lämnar Android Wear applikationsdelen så slutar datan att överföras mellan smarttelefonen och smartklockan, för att anslutningen mellan dem bryts. Jag var tvungen att programmera i applikationen så att skärmen och applikationen förhindras från att slockna och att applikationen skulle fortsätta skicka data. Det gjorde jag med en Wakelock klass som finns i Android APIn. Det leder till att man inte kan använda smartklockan till något annat och den använder mera batteri än om den skulle fortsätta att skicka data med släckt skärm. Dataöverföringen avbryts om skärmen rörs av misstag och smartklockan registrerar det som en svepning från höger till vänster, vilket resulterar i att applikationen stängs av.

5 TEST AV SENSORERNAS NOGGRANNHET

I detta kapitel kommer jag gå igenom testet som jag gjorde för att se hur noggranna Android Wears sensorer är. Jag gjorde tre olika test: en som borde ha låg hjärtpuls, en som borde ha normal hjärtpuls och en som borde ha hög hjärtpuls.

5.1 Låg hjärtpuls test

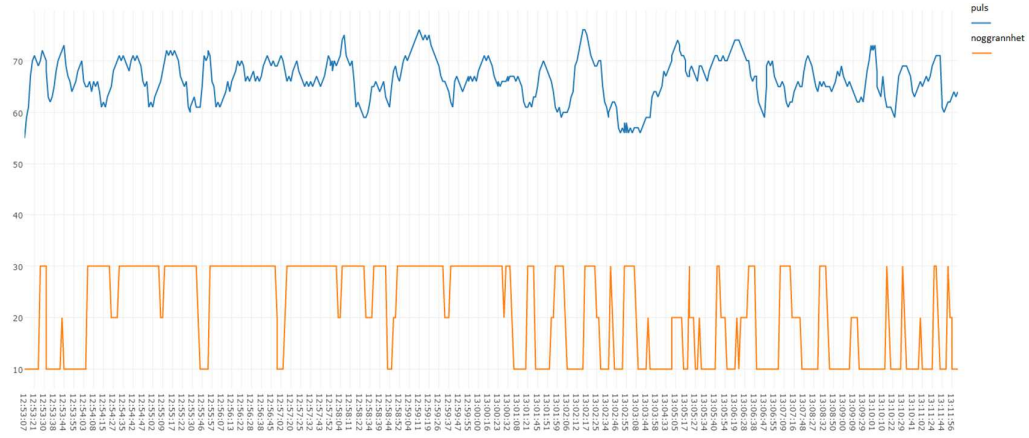
Jag gjorde testet där jag skulle ha låg hjärtpuls genom att ligga på en säng ca en halv timme för att samla upp data. Från grafen nedan kan man utläsa hjärtpuls värdena som Android Wear smartklockan uppmätte. Till största delen ligger min puls mellan 60 och 70 slag per minut och ibland stiger den upp till ca 80, 84 slag per minut som högst. Den blåa linjen är pulsen. Den gula linjen är noggrannheten på mätningen enligt Android Wears sensor som kan vara mellan 1 och 3. Jag har ändrat 1 till 10, 2 till 20 och 3 till 30 så att man bättre ser variationen i noggrannheten.



Figur 24. Mätningsresultat för låg hjärtpuls

5.2 Normal hjärtpuls test

Testet där jag skulle ha normal puls utfördes genom att jag satt på en stol ca en halv timme. Min puls som uppmättes under detta test var mellan 60 och 75 slag per minut enligt grafen i Figur 25. Normal vilopuls för en vuxen person är mellan 60 och 80 slag per minut [24],så Android Wear smartklockans sensor mätningar verkar vara korrekta. Noggrannhetens skala har ändrats liksom i Figur 24.

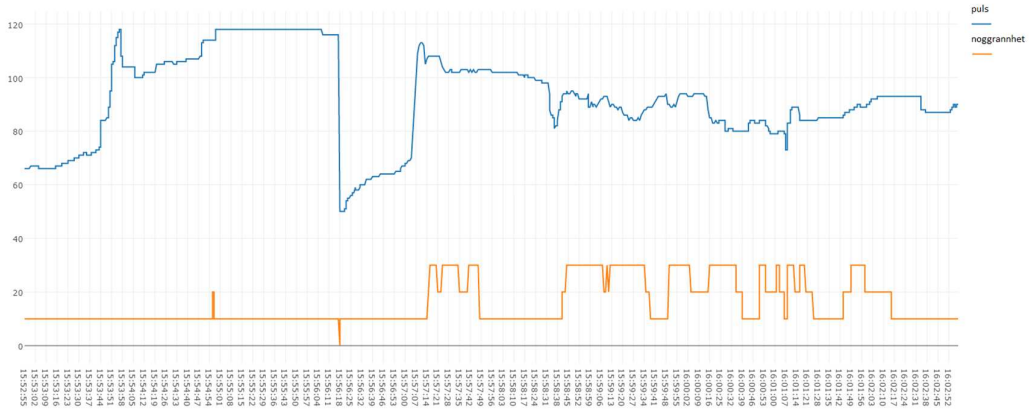


Figur 25. Mätningsresultat för normal hjärtpuls

5.3 Hög hjärtpuls test

För att uppnå hög hjärtpuls gjorde jag några övningar för att höja min hjärtpuls. Jag började med att springa på stället, sedan gjorde jag så många knäböj som jag orkade och till sist gjorde jag armpressar.

Som man kan utläsa från grafen är hjärtpulsen normal till en början 68-70 slag per minut och stiger sedan kraftigt till 100-115 slag per minut under träningen, det märkliga hoppet ner till 50 slag per minut måste vara någon felläsning från Android Wears sida, för jag hade smartklockan på armen under hela testets gång. Android Wears mätningar är kanske inte alltid helt pålitliga. Noggrannhetens skala har ändrats liksom i Figur 24.



Figur 26. Mätningsresultat för hög hjärtpuls

5.4 Slutsatser från testerna

De uppmätta värdena i testerna med låg och normal puls var på ganska samma nivå, 60-70 hjärtslag per minut, men normal puls testet gav jämnare värden än låg puls testet. Normal puls testet hade också högre noggrannhet. Skillnaden i mätresultaten mellan normal puls testet och hög puls testet var märkbara. Pulsen var på en betydligt högre nivå, upp till 115 slag per minut.

6 SLUTSATSER

Jag valde att göra mitt examensarbete om att skapa en applikation för Android och Android Wear för att jag tänkte att det skulle vara intressant att skapa en Android Wear applikation, eftersom Android Wear var väldigt nytt när jag började med projektet. Det visade sig vara både intressant och utmanande. Jag har tidigare skapat liknande server/databaser, och applikationer för Android, men jag har inte utvecklat en applikation för Android Wear. Det tog också tid att vänja sig vid den nya utvecklingsmiljön Android Studio.

Slutprodukten blev en fungerande prototyps applikation trots att den inte fungerade som jag hade tänkt från början p.g.a. att dataöverföringen mellan smartklockan och smarttelefonen avbryts då smartklockans skärm släcks. Detta problem åtgärdades med hjälp av Wakelock (som är en klass som finns i Androids API). Detta är inte en bra lösning eftersom man inte kan använda smartklockan till annat då man tar upp data och smartklockans batteri förbrukas snabbare.

Man kunde vidareutveckla applikationen så att när smarttelefonen inte har kontakt med servern skulle datan sparas lokalt på smarttelefonen och när den får kontakt med servern igen skulle smarttelefonen skicka datan till servern. Alternativt kunde datan sparas lokalt på smarttelefonen och vid varje halv timme eller timme skickas till servern för att spara på smarttelefonens batteri då den inte konstant behöver överföra data till servern.

KÄLLOR

[1] Willard Cody, *The Wearables Revolution will be the biggest market ever*, Market Watch Tillgängligt: http://blogs.marketwatch.com/cody/2014/08/07/the-wearables-revolution-will-be-the-biggest-market-ever/, Hämtad 15.5.2016

[2] Wikipedia, 2016, *List of mergers and acquisitions by Google*, Tillgänglig: https://en.wikipedia.org/wiki/List_of_mergers_and_acquisitions_by_Google, Hämtad: 14.5.2016

[3] Wikipedia, 2016, *Android*, Tillgänglig: https://en.wikipedia.org/wiki/Android_(operating_system), Hämtad: 14.5.2016

[4] Android Developer, 2016, *Platform Versions*, Android Developer. Tillgänglig: https://developer.android.com/about/dashboards/index.html?utm_source=suzunone, Hämtad: 21.3.2016

[5] Gergenta Marko, Nakamura Masumi. 2014, *Learning Android*, 2 uppl., Californien O'Reilly Media, Inc., 288 s

[6] Wikipedia, 2015, *Virtual machine*, Tillgänglig: https://en.wikipedia.org/wiki/Virtual_machine, Hämtad: 18.3.2016

[7] Wikipedia, 2015, *Android Runtime*, Tillgänglig: https://en.wikipedia.org/wiki/Android_Runtime, Hämtad: 11.11.2015

[8] Nardone Massimo, Silva Vladimir. *Pro Android Games*, 3 uppl., New York City Apress Media LLC 395s

[9] Android Developer, 2016, *Package Index*, Android Developer. Tillgänglig: http://developer.android.com/reference/packages.html, Hämtad: 4.5.2016

[10] Android Developer, 2016, *Sensor Event*, Tillgänglig: http://developer.android.com/reference/android/hardware/SensorEvent.html, Hämtad: 4.5.2016

[11] Android Developer, 2016, *Sensor Manager*, Tillgänglig: http://developer.android.com/reference/android/hardware/SensorManager.html, Hämtad: 4.5.2016

[12] Amadeo Ron, 2015, *Moto 360 review—Beautiful outside, ugly inside* 8.9.2014, Ars Technia. Tillgänglig: http://arstechnica.com/gadgets/2014/09/moto-360-review-beautiful-outside-ugly-inside/, Hämtad: 20.1.2015

[13] Nickinson Phil, 2015, *LG G Watch R specs 27.8.2014*, Android Central. Tillgänglig: http://www.androidcentral.com/lg-g-watch-r-specs, Hämtad: 20.1.2015

[14] Swider Matt, 2015, *Samsung Gear Live review 13.8.2014*, Tech Radar. Tillgänglig: http://www.techradar.com/reviews/phones/mobile-phone-accessories/samsung-gear-live-review-1256537/review, Hämtad: 20.1.2015

[15] Wikipedia, 2015, *Electrocardiography*, Tillgänglig: https://en.wikipedia.org/wiki/Electrocardiography, Hämtad: 20.11.2015

[16] Reisner Andrew, Shaltis Phillip A., McCombie Devin, Asada H. Harry. 2008, Utility of the Photoplethysmogram in Circulatory Monitoring. *Anesthesiology*, vol. 108, s. 950-958

[17] Woodford Chris, *How do pedometers work?*, Explain That Stuff!. Tillgänglig: http://www.explainthatstuff.com/how-pedometers-work.html, Hämtad: 13.8.2015

[18] *FAQ Where did Eclipse come from?*,2015 Eclipse Organisation. Tillgänglig: http://wiki.eclipse.org/FAQ_Where_did_Eclipse_come_from%3F, Hämtad: 6.7.2015

[19] Protalinski Emil, *Google releases Android Studio 1.0, the first stable version of its IDE 8.12.2014*, Venturebeat. Tillgänglig: http://venturebeat.com/2014/12/08/GOOGLE-RELEASES-ANDROID-STUDIO-1-0-THE-FIRST-STABLE-VERSION-OF-ITS-IDE/, Hämtad: 21.4.2015

[20] Tomcat Organisation, 2016 Tillgänglig: http://tomcat.apache.org/, Hämtad: 19.3.2016

[21] Tomcat Organisation, 2016, *Powered By*, Tomcat Organisation. Tillgänglig: http://wiki.apache.org/tomcat/PoweredBy, Hämtad: 19.3.2016

[22] *What is REST?*, Learn REST: A Tutorial. Tillgänglig: http://rest.elkstein.org/, Hämtad: 19.3.2016

[23] JSON Organisation, 2015. *Introducing JSON*. Tillgänglig: http://json.org/ Hämtad: 5.7.2015

[24] Wikipedia, 2016, *Sydämen syke*, Tillgänglig: https://fi.wikipedia.org/wiki/Syd%C3%A4men_syke, Hämtad: 28.4.2016